

Optimized network clustering by jumping sub-optimal dendrograms

Nicolas Bock*

Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

Erik Holmström

*Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA and
Instituto de Física, Universidad Austral de Chile, Casilla 567, Valdivia, Chile*

Johan Brännlund

Department of Mathematics & Statistics, Dalhousie University, Halifax, NS B3H 3J5 Canada

(Dated: July 2, 2009)

We propose a method to improve community division techniques in networks that are based on agglomeration by introducing dendrogram jumping. The method is based on iterations of sub-optimal dendrograms instead of optimization of each agglomeration step. We find the algorithm to exhibit excellent scaling behavior of its computational complexity. In its present form the algorithm scales as $\mathcal{O}(N^2)$, but by using more efficient data structures it is possible to achieve a scaling of $\mathcal{O}(N \log^2 N)$. We compare our results with other methods such as the greedy algorithm and the extremal optimization method. We find modularity values larger than the greedy algorithm and values comparable to the extremal optimization method.

I. INTRODUCTION

The study of communities in networks has received considerable attention recently. Generally, a community can be thought of as a subset of nodes of the network in which the nodes within a community are more connected among each other than they are connected to the other nodes in the network. By analyzing a network in terms of its communities, it is possible to gain understanding of the structure of a network on a larger scale and to uncover previously unnoticed connections between nodes or groups of nodes. Examples of successful community division studies include a study on the relationship between diseases and genes [1], the identification of transition states in potential energy landscapes [2], and the identification of recording locations and racial community structures in a jazz musicians network in the USA around 1920 [3]. We have recently used our modularity optimization algorithm to optimize the performance of a recursive inverse factorization technique used in large scale electronic structure calculations [4].

The analysis of a network in terms of communities poses a difficult challenge. The clustering algorithm has to be accurate so that it identifies informative community structures. This implies that the algorithm has to consider many of the possible community divisions in the network before it can decide which one is the best. From a computational point of view we are faced with a rapidly growing problem as a function of network size. To consider all possible community divisions becomes computationally unfeasible even for relatively small networks. In fact, it can be shown that the number of ways of dividing a network into communities grows as the Sterling

number of the second kind [5].

In order to quantify the quality of a community division, a quality function is introduced that assigns a “value-reflecting” number, or “quality-of-split” [6] to a community division. The modularity, as introduced by Girvan and Newman [7], is a popular choice for such a quality function. Although there may be some drawbacks to this approach as pointed out by Fortunato and Barthélemy [8], the community divisions that are obtained by optimizing the modularity typically give valuable information [1, 2].

In the literature many different optimization strategies can be found that employ the modularity. They vary in quality, i.e. the value of the largest modularity they find, and in the computational effort. An efficient agglomerative method is the greedy algorithm of Newman [5] which Clauset et al. [9] showed to run in $\mathcal{O}(N \log^2 N)$ computational effort. Other methods however, find modularity values larger than the greedy algorithm. These include extremal optimization [10], basin-hopping [2], simulated annealing [11], recursive filtration [12], a heuristic algorithm [13], and a spectral algorithm [14].

In this article we introduce a new agglomerative method which we demonstrate by using the modularity as the quality-of-split function. Our method is general however and can be used with any other quality-of-split function. We find modularity values comparable to the extremal optimization method for a set of well-studied networks. In section II we summarize the agglomerative greedy algorithm and introduce our method. In section III we compare with results for some popular networks using the greedy method and the extremal optimization method. Finally, in section IV we present our conclusions.

*Electronic address: nbock@lanl.gov

II. THEORY

A network of N nodes can be divided into any number of communities, C , where $1 \leq C \leq N$. The extremal cases $C = 1$ and $C = N$ are the two trivial solutions in which all nodes either belong to only one common community or each belongs only to its own separate community, respectively. Given that the number of possible community divisions is exceedingly large for any decently sized network, the problem of finding the optimal community split cannot be approached by calculating all possible splits. Instead one has to resort to approximate solutions of the problem. One possibility is to attempt to find the optimal community division by starting with one of the two trivial cases and proceeding by either stepwise merging two communities or by splitting a community into two until the opposite extremal case is reached. These two approaches are commonly referred to as agglomerative and divisive methods, respectively. The set of community divisions for every value of C between $1 \leq C \leq N$ is called a dendrogram. Such dendrogram-based optimization methods aim to find the best community split by optimizing each step along the dendrogram based on the change in the quality-of-split measure [5, 15, 16]. These methods are typically very efficient computationally, but the quality of their result depends critically on the heuristic chosen during the stepwise optimization process. In fact, in a recent comparison of dendrogram-based methods and a simulated annealing technique, Danon et al. [17] found that the simulated annealing method, which is not bound to a dendrogram, is able to find better solutions to the community division problem than the dendrogram-based methods.

The quality-of-split function we will use for this study is the modularity, Q , of Girvan and Newman [7]. It is defined as

$$Q = \text{Tr}(e) - \sum_{ij} (e^2)_{ij}, \quad (1)$$

where e is the assortative mixing matrix (which is a $C \times C$ matrix). The elements e_{ij} are given by the number of links from community i to j for a particular community division as a fraction of the total number of links.

The greedy algorithm was introduced by Newman [5] and is a good example of an agglomerative method. A range of other agglomerative methods have been proposed which differ in how each step in the merge process is chosen [5, 6, 9, 18]. The change in modularity due to merging two communities i and j can be written as [5]

$$\Delta Q = e_{ij} + e_{ji} - a_i b_j - a_j b_i, \quad (2)$$

where a_i and b_i are the column and row sums of e , respectively. Each merge process is chosen such as to maximize the effected modularity change in the hope that this leads to the community division with the maximum

modularity. What makes the greedy method particularly attractive is the fact that eq. (2) is inexpensive to evaluate (it is of $\mathcal{O}(1)$ computational effort). In addition, the number of community merges to evaluate is at most $(N-1)(N-2)\dots$ which leads to an overall computational complexity of $\mathcal{O}(N \log^2 N)$ [9].

A. Sub-optimal iterations

Optimization techniques which are bound to a dendrogram generally find smaller values of the modularity than methods which do not operate along a dendrogram [19]. This is due to the fact that the optimal solution may not be accessible by walking a stepwise optimized dendrogram. Optimal choices in the beginning of the dendrogram (large C for agglomerative methods) may lead to community divisions which can not be merged further to achieve the optimal division. This is of course also a problem for divisive methods. It may therefore not be possible to find a heuristic that finds the optimal community division along one single dendrogram. Motivated by our previous work on the modularity density of networks [20] we avoid the problem by performing incomplete optimizations in each merge process. We do not consider all $C(C-1)$ possible merges, but only a smaller randomly chosen subset and pick the merge with the largest modularity increase from this subset. This procedure has the advantage that it allows for randomness in the agglomeration process so that two different runs will not give the same result. The randomness is obviously also a drawback since it is very unlikely that we find the optimal community split in one such random dendrogram. Even several sub-optimal dendrograms will be unlikely to have produced the optimal community division. We therefore iterate over several random dendrograms in an inner loop and optimize a list of modularity values for each value of C in an outer loop. Our algorithm is expressed in terms of a pseudocode in figure 1.

Inside the inner loop we analyze a “somewhat” random dendrogram. By this we mean that we pick the best out of n randomly chosen possible merge processes in each step of the dendrogram (line 4 in figure 1). Clearly, in the limit $n \rightarrow 1$, the dendrogram is random. The greedy method, on the other hand, corresponds to at most $n = C(C-1)$ different merge processes. For any value $n < C(C-1)$ we achieve sub-optimal agglomeration. As we decrement C and walk the dendrogram from larger to smaller C values, the merge process with the largest modularity gain becomes the proposed merge process. We store a list of the best modularities found so far for each value of C and merge along the proposed merge process only if the proposed merge produces a higher modularity value than the previous best value. If the proposed merge process produces a modularity value equal or less than the best modularity value thus far, we discard it and load the community division corresponding to the best modularity found so far to continue. It is

```

1: for  $N^{\text{outer}}$  times do {Outer loop}
2:   for  $N^{\text{inner}}$  times do {Inner loop}
3:     for  $C = N$  to  $C \geq 2$  do {Suboptimal agglomeration}
4:       Find  $n$  random merge candidates.
5:       Calculate  $\Delta Q_n(C \rightarrow C-1)$  from eq. (2).
6:       if  $\max(Q_C^{\text{inner}} + \Delta Q_n) > Q_{C-1}^{\text{inner}}$  then {Accept proposed
         merge process}
7:          $e_C^{\text{inner}} \leftarrow e$ 
8:          $Q_{C-1}^{\text{inner}} \leftarrow Q_C^{\text{inner}} + \Delta Q_n$ 
9:         Calculate new  $e$ .
10:      else {Reject proposed merge process}
11:       $e \leftarrow e_{C-1}^{\text{inner}}$ 
12:    end if
13:  end for{Suboptimal agglomeration}
14: end for{Inner loop}
15: for  $C = 1$  to  $C < N$  do {Update list of modularity val-
    ues}
16:   if  $Q_C^{\text{inner}} > Q_C^{\text{outer}}$  then
17:      $Q_C^{\text{outer}} \leftarrow Q_C^{\text{inner}}$ 
18:   end if
19: end for{Update list of modularity values}
20: end for{Outer loop}

```

FIG. 1: Sub-Optimal Dendrogram Jumping Algorithm: (line 3) This loop performs the suboptimal agglomeration. In this study Q means the modularity, but any quality-of-split function can be used. (line 2) The inner loop restarts the agglomeration process. We improve on the best modularities found by accepting proposed merge processes only if they lead to higher modularity values. (line 1) The best modularity values found in the inner loop are stored and the inner loop is restarted.

this step (lines 6 through 12 in figure 1) which leads to a dendrogram jump. The list of best modularity values of the inner loop usually converges rather quickly and the algorithm cannot improve on the modularities any longer.

In the outer loop we store another list of best modularity values. Once the inner loop completes we update the list of the outer loop with the values found in the inner loop (lines 15 through 19 in figure 1). We then reinitialize the inner loop for another run. This step allows the algorithm to explore a different part of the community division space since it will randomly choose other dendrograms.

The 3 tunable parameters in our algorithm, n , N^{inner} , and N^{outer} , are chosen according to the following heuristic: (1) The number of suboptimal trials, n , should be significantly smaller than the greedy limit, $C(C-1)$. Although our method will work with any $n \geq 1$, we have found in practice that any value between $5 \leq n \leq 10$ works well. (2) The number of inner loops, N^{inner} , should be chosen as small as possible such that the Q_C^{inner} list is converged. This process is exemplified by the convergence of the blue dashed lines in the upper panel of figure 2. The value of this particular parameter depends strongly on the network size. We have found values of N^{inner} between 8 and 150 for the smaller networks (Zachary Karate Club) and the larger networks (Jazz mu-

sician and e-mail network), respectively, to work well. (3) There is no predetermined limit on the number of outer loops, N^{outer} . It is determined by the level of convergence desired for the maximum modularity value. The more outer loops are performed, the more likely it is to find the maximum modularity value.

Iterating over the inner and outer loop converges rapidly and we find a list of the best modularity values for all C . In the following section we will compare our results with previously studied networks.

III. RESULTS

In the upper panel of figure 2 we show the best modularity values found by our iterative dendrogram jumping method for the unweighted Zachary Karate Club network (black circles). This result was obtained with $N^{\text{inner}} = 8$ inner loop iterations and $N^{\text{outer}} = 20$ outer loop iterations. The number of sub-optimal trials in each agglomeration was $n = 10$. We indicate the borders between sections of the curve that belong to the same dendrogram to illustrate the dendrogram jumping of our algorithm. In the case shown, we find 12 such borders. We find the maximum modularity at $C = 4$ at a value of $Q = 0.4198$. The blue dashed lines show the evolution of an optimization in the inner loop consisting of 8 sub-optimized $Q(C)$ curves obtained by 10 merge trials per C . We find that this particular run over the inner loop of the dendrogram jumping method achieves high modularities for $C = 3, 7$, and 8 but fails to find the largest modularity that we find after looping over the outer loop.

In the lower panel of figure 2 we show the first few branches of the resulting disconnected dendrogram for the overall optimization of the Zachary Karate Club network. At $C = 1$, all nodes are in the same community as indicated by the single black circle. As this community is split into two and subsequently three communities, we find that the figure looks like an ordinary dendrogram. However, splitting 3 into 4 communities, the community division corresponding to the highest modularity value results in the merging of two communities and the splitting of two communities. We indicate this merging by the first red circle. This is a situation that cannot occur in a dendrogram and thus indicates a dendrogram jump in the maximum modularity curve. Two more such events are marked with red circles.

We have performed modularity optimizations by means of our improved sub-optimal dendrogram jumping algorithm for a set of well-known networks. In table I we present our results for the maximum modularity and the number of communities that were obtained. The results are compared to the corresponding maximum modularity found by the greedy method [14] and the extremal optimization method [10], where applicable. We find that dendrogram jumping always finds modularity values larger than the greedy algorithm and values comparable to the extremal optimization method. In most cases

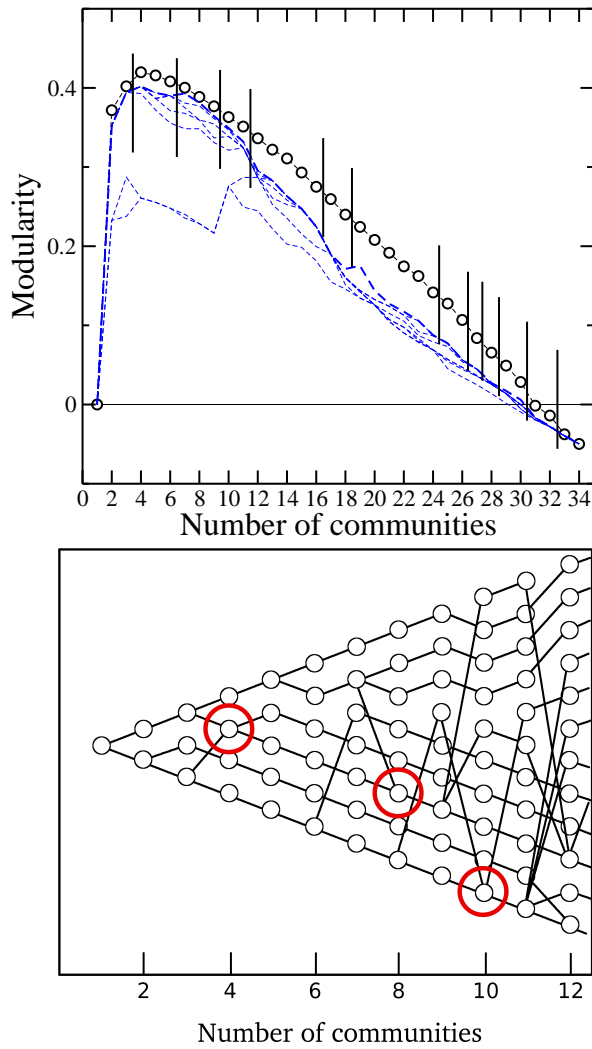


FIG. 2: **Upper panel:** The dashed lines (blue online) indicate the maximum modularity values found in the inner loop. With each iteration of the inner loop the maximum modularity increases and eventually converges, shown with the bold dashed line (blue online). The circles show the maximum modularity values found after the outer loop is converged. The vertical lines indicate sections that belong to one dendrogram. Dendrogram jumping occurs across the vertical lines. **Lower panel:** A generalized dendrogram that corresponds to the maximum modularity of the unweighted Zachary Karate Club network. Events that indicate dendrogram jumps are marked with red circles.

the number of communities corresponding to the largest modularity value differs from what was found with the other methods. We never found a case in which the communities had the same members in the different methods even in the cases where the number of communities was the same. For the small networks the community members differed only in few nodes. For the larger networks however, we found more significant differences in the assignment of nodes to communities.

The evaluation of ΔQ of eq. (2) can be done in computational effort $\mathcal{O}(1)$ time. The subsequent update of e after a merge operation takes $\mathcal{O}(N)$ worst time and there are $N - 1$ such merge operations per dendrogram. We iterate through a fixed number of inner and outer loops, i.e. the total computational effort is $\mathcal{O}(N^2)$. Not surprisingly, this is identical to the computational effort found for the greedy algorithm [5] in the sparse graph limit. The inner loop of our algorithm is a generalization of the greedy method in the limit of $n \rightarrow C(C - 1)$. In our algorithm we only consider a small fixed number n of merge candidates, which implies that our method always scales $\mathcal{O}(N^2)$ even in the dense graph case. However the data structures used by Clauset et al. [9] to speed up the greedy algorithm clearly are readily applicable in our case as well, which makes it possible to reduce the computational effort to $\mathcal{O}(N \log^2 N)$. For comparison, the extremal optimization [10] method runs in $\mathcal{O}(N^2 \log N)$ time.

We have successfully used our algorithm in its current implementation for networks of up to ≈ 1200 nodes but networks an order of magnitude larger should be analyzable with a current desktop workstation and sufficient memory installed.

IV. CONCLUSIONS

We have presented a generic optimization technique that applies to community detection algorithms that are agglomerative. We demonstrated the efficiency of our method by calculating the maximum modularity for a set of networks and comparing our results with two other methods, the greedy method by Newman and coworkers [5, 9] and the extremal optimization method by Duch and Arenas [10]. In this comparison we found modularity values for all examples studied that are larger than the results of the greedy algorithm and comparable to the results of the extremal optimization method. The computational complexity of our method ultimately is $\mathcal{O}(N \log^2 N)$. Our method therefore has the same computational complexity scaling behavior as the greedy method. The extremal optimization method is computationally more expensive and scales with a computational complexity of $\mathcal{O}(N^2 \log N)$. In applications in which sufficient memory is available, our algorithm therefore is the method of choice.

Our study showed that the community divisions of maximum modularity are not connected by one single dendrogram and thus any method which aims at optimizing the modularity by optimizing each step along a dendrogram will fail. This finding confirms a similar conclusion drawn by Medus et al. [19]. This has important implications for the further development of modularity optimization methods.

Network	Ref.	Nodes	Edges	Q^{\max}	C	Q^{greedy}	C^{greedy}	Q^{EO}	C^{EO}
Zachary	21, 22	34	156	0.4198	4	0.3807	3	0.4188	4
Zachary (W)	21, 22	34	156	0.4449	4	0.4345	3		
Fraternities Subj. (W)	21, 23, 24	58	3306	0.0486	3	0.0412	3		
Fraternities Obj. (W)	21, 23, 24	58	1934	0.1460	6	0.1408	6		
Dolphins	25, 26	62	318	0.5285	5	0.4923	4		
Prisoners	21, 27	67	182	0.6232	9	0.6217	9		
Les Miserables (W)	21, 28	77	508	0.5667	6	0.5472	5		
Les Miserables	21, 28	77	508	0.5600	6	0.5006	5		
Grassland	29	88	274	0.6627	9	0.6609	10		
Jazz bands	3, 30	198	5484	0.4450	4	0.4389	4	0.4452	5
Littlerock	29	183	4886	0.3629	4	0.3395	3		
Jazz musicians	3, 30	1265	76714	0.5780	18	0.5235	20		
e-mail	31, 32	1133	10902	0.5718	11	0.5093	15	0.5738	15

TABLE I: Our results for the optimized networks in this study compared to the greedy algorithm and the extremal optimization method were applicable. The entries labeled (W) are weighted networks. Shown are the number of nodes in the network (Nodes), the number of directed edges (Edges), the maximum modularity found by our method (Q^{\max}), the number of communities for this value of the modularity (C), the same quantities for the greedy algorithm (Q^{greedy} and C^{greedy}), as well as for the extremal optimization method (Q^{EO} and C^{EO}).

V. ACKNOWLEDGMENTS

We would like to thank Travis Peery, Eric Chisolm, and Anders Niklasson for many helpful discussions. We would also like to thank Mark Newman who graciously

provided us with his network data which we analyzed in this paper. Special acknowledgments also to the genuine and professional scientific atmosphere provided by the International Ten Bar Café.

-
- [1] D. Wilkinson and B. A. Huberman, PNAS **101**, 5241 (2004).
 - [2] C. P. Massen and J. P. K. Doye, Phys. Rev. E **71**, 046101 (2005).
 - [3] P. M. Gleiser and L. Danon, Advances in Complex Systems **6**, 565 (2003).
 - [4] E. H. Rubensson, N. Bock, E. Holmström, and A. M. N. Niklasson, J. Chem. Phys. **128**, 104105 (2008).
 - [5] M. E. J. Newman, Phys. Rev. E **69**, 066133 (2004).
 - [6] J. H. Ward, Journal of the American Statistical Association **58**, 236 (1963).
 - [7] M. Girvan and M. E. J. Newman, PNAS **99**, 7821 (2002).
 - [8] S. Fortunato and M. Barthélemy, PNAS **104**, 36 (2006).
 - [9] A. Clauset, M. E. J. Newman, and C. Moore, Phys. Rev. E **70**, 066111 (2004).
 - [10] J. Duch and A. Arenas, Phys. Rev. E **72**, 27104 (2005).
 - [11] C. O. Dorso, A. Medus, and G. Acuna, Physica A **358**, 593 (2005).
 - [12] Y. Shen, W. Pei, K. Wang, T. Li, and S. Wang, Physica A: Statistical Mechanics and its Applications **387**, 6663 (2008).
 - [13] D. Chen, Y. Fu, and M. Shang, Physica A: Statistical Mechanics and its Applications **388**, 2741 (2009).
 - [14] M. E. J. Newman, PNAS **103**, 8577 (2006).
 - [15] M. Gustafsson, M. Hornquist, and A. Lombardi, Physica A **367**, 559 (2006).
 - [16] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
 - [17] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, Journal of Statistical Mechanics: Theory and Experiment **2005**, P09008 (2005).
 - [18] J. MacQueen, In: Le Cam L.M., Neyman J. (Eds.), Proc. 5th Berkley Symp on Mathematical Statistics and Probability **1**, 666 (1967).
 - [19] A. Medus, G. Acuna, and C. O. Dorso, Physica A **358**, 593 (2005).
 - [20] E. Holmström, N. Bock, and J. Brännlund, Physica D: Nonlinear Phenomena **238**, 1161 (2009).
 - [21] <http://vlado.fmf.uni-lj.si/pub/networks/data/UciNet/UciData>
 - [22] W. W. Zachary, J. Anthropol. Res. **33**, 452 (1977).
 - [23] H. Bernard, P. Killworth, and L. Sailer, Social Networks **2**, 191 (1980).
 - [24] H. Bernard, P. Killworth, and L. Sailer, Social Science Research **11**, 30 (1982).
 - [25] <http://www-personal.umich.edu/~mejn/netdata/>.
 - [26] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, Behavioral Ecology and Sociobiology **54**, 396 (2003).
 - [27] J. MacRae, Sociometry **23**, 360 (1960).
 - [28] D. E. Knuth, *A platform for combinatorial computing*, The Stanford GraphBase, Addison-Wesley, Reading, MA (1993).
 - [29] <http://www.foodwebs.org/>.
 - [30] Data obtained through private communication.
 - [31] R. Guimera, L. Danon, A. Diaz-Guilera, F. Giralt, and A. Arenas, Phys. Rev. E **68**, 065103(R) (2003).

[32] Data obtained through private communication.